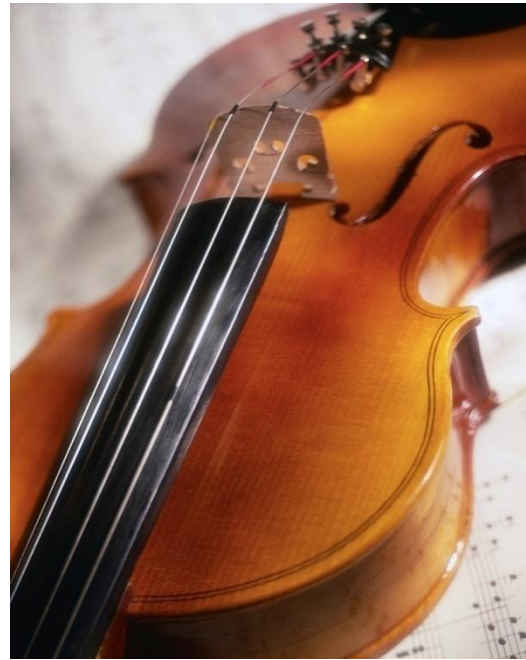


CHARACTERS AND STRINGS

CSSE 120—Rose Hulman Institute of Technology

Characters and Strings



Characters in Python

- Just a special case of *string*

```
>>> myChar = 'C'
```

```
>>> print myChar
```

```
C
```

```
>>> print ord(myChar) # converts character to int
```

```
67
```

```
>>> print chr(67) # converts int to character
```

```
C
```

Characters in C

- C's **char** type is really a kind of number!
- A **char** takes 1 byte of storage space
- Example:

```
char myChar;  
myChar = 'C';  
printf("%c\n", myChar); /* %c is format spec. for char */  
printf("%d\n", myChar); /* can print char as a decimal */  
printf("%c\n", 67);    /* can print int as char */  
myChar++;  
printf("%c\n", myChar); /* What prose do you suppose? */
```

Ten Ways to Say 'A'

```
char c = 'A';
int i = 'A';
printf("A");
printf("%c", 'A');
printf("%c", 'B'-1);
printf("%c", c);
printf("%c", i);
putchar('A'); /* can "push" single characters to output */
putchar('C' - 2);
putchar(toupper('a')); /* Need to #include <ctype.h> */
putchar(c);
putchar(i);
```

Math with Characters

- We can do math with character types:
 - `'C' + 1 == 'D'`
 - `char b = 'b';`
`b--;`
`putchar(b); /* outputs a */`
- Combine these ideas to write a **for** loop that prints the characters from 'a' to 'z' on a single line
 - Try this in Eclipse, work with a neighbor
 - Write your answer on your quiz

Getting Characters

- To read a single character from the console use:
 - **getchar()**
 - Caveat: **getchar()** returns an **int**, either a **char** value or **EOF** (end of file)

```
void getSomeChars() {  
    int inChar;  
    int count = 0;  
    printf("\n\nType some text, then press 'Enter': ");  
    fflush(stdout);  
    inChar = getchar();  
    while (inChar != '\n' && inChar != EOF) {  
        count++;  
        inChar = getchar();  
    }  
    printf("\nYou entered %d characters.", count);  
}
```

Note: most operating systems only pass characters to your program after the user presses the **enter** key

Character Functions: *ctype.h*

- Conversion Functions:
 - ▣ `int tolower(int c);`
 - ▣ `int toupper(int c);`
- Test functions:
 - ▣ `isdigit(c)`
 - ▣ `isalpha(c)`
 - ▣ `islower(c)`
 - ▣ `isupper(c)`
 - ▣ `isspace(c)`

See the *C Library Reference* link on ANGEL under Course Resources for more functions.

Just Stringing You Along

- "Strings" in C are just
 - ▣ arrays of characters,
 - ▣ with a '\0' at the end

- Examples:

- ▣ **char fname[] = "Lou";** **char lname[10];**

...note difference in box-and-pointer diagrams

- How would we assign "Gehrig" to lname?
 1. char lname[] = "Gehrig"
 2. character-by-character assignment
 3. strcpy(coming soon)

String variables vs. constants

- String Variable

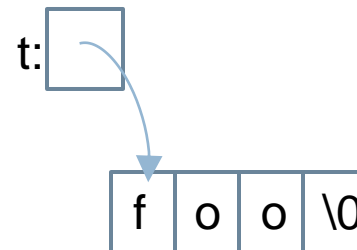
- `char s[] = "foo";`



- String Constant

- `char *t = "foo";`

- Strings declared in this way **cannot** be mutated!



String Functions: *string.h*

Function	Purpose
<code>char *strcpy(char *dest, char* src)</code>	copy string src to string dest, including '\0'; return dest Note: strings are <i>mutable</i> in C, unlike Python!
<code>char *strcat(char *dest, char* src)</code>	concatenate string src to end of dest; return dest
<code>int strcmp(char *str1, char *str2)</code>	compare string str1 to string str2, return a negative number if $str1 < str2$, zero if $str1 == str2$, or positive otherwise
<code>char *strstr(char *str1, char *str2)</code>	return a pointer to first occurrence of str2 in str1, or NULL if not present
<code>size_t strlen(char *str)</code>	return length of str (size_t is a typedef for int on most systems)

Note: we usually ignore the return stypes on strcpy and strcat, since they mutate dest.

Descriptions from K&R, p. 249.
See the *C Library Reference* link on ANGEL for more.

String Concatenation Using *strcat()*

- Consider:

```
char s1[] = "Go, Red! Go, White! ";  
char s2[] = "Go Rose, Fight!";  
/* ??? */  
printf("%s\n", s3);
```

- What goes in the space? We want:

- the output to be

Go, Red! Go, White! Go Rose, Fight!

- and no additional string literals

Summary: Strings in C

- Strings are arrays of characters:

- ▣ `char fname[] = "Lou";`

or

- ▣ `char lname[10];`
`strcpy(lname, "Gehrig");`

- "Null terminated", that is, a `'\0'` at the end
- Don't forget to reserve enough space to hold the string
 - ▣ (next session we'll see how to ask for just enough space dynamically)



When C Gives You Lemons...

- Problem:
 - ▣ Python includes high level functions for strings
 - ▣ C (and some other languages) do not
 - ▣ What if you need to use C, but also need strings?
- Solution: Make your own string functions!
- Homework:
 - ▣ Check out ***Session26CharactersAndStrings*** from SVN
 - ▣ See homework description linked from ANGEL
 - ▣ Let's start it together.